Self-Hosted GGUF LLM Chatbot – Complete End-to-End Documentation

> **Goal** Build, test locally, and deploy online a **fully private AI chatbot** using the `mistral-7b-instruct.Q4_K_M.gguf` model, hosted **inside your backend**, integrated with your website (Lovable frontend), and capable of answering questions using **RAG (Retrieval-Augmented Generation)** from documents you provide.

This document covers **everything end-to-end**: - Where & how to download the model - Local setup and testing - Backend integration with GGUF - RAG (learning from documents you provide) - Moving from local → online server - Connecting the hosted model to your website - Maintenance & future upgrades

---

1. Final Architecture (What We Are Building)

```
Lovable Website (Frontend)
        ↓ HTTPS
Backend API (FastAPI)
        ↓
llama.cpp (Python binding)
        ↓
GGUF Model (Mistral 7B Instruct)
        ↓
FAISS Vector DB (RAG)
        ↓
Your Documents / Knowledge Files
```

Key Principles

- Model runs **inside your backend**
- No Ollama, no SaaS, no third-party APIs
- Frontend never touches the model
- All learning happens via **RAG**, not unsafe retraining

---

2. Model Selection (Fixed & Confirmed)

Model We Are Using

```
mistral-7b-instruct.Q4_K_M.gguf
```

Why This Model

- Free & open
- Optimized for CPU (no GPU required)
- Fits in 8–16 GB RAM
- Stable for chat & support use-cases
- Works perfectly with `llama.cpp`

This file becomes the **core brain** of your chatbot.

---

3. Downloading the Model (Very Important)

Official Source

Models are hosted on **Hugging Face** by trusted maintainers.

Recommended Repository

- TheBloke / Mistral-7B-Instruct-GGUF

File to Download

```
mistral-7b-instruct.Q4_K_M.gguf
```

Download Command (Linux / Server)

```
mkdir -p ai-backend/models
cd ai-backend/models
wget https://huggingface.co/TheBloke/Mistral-7B-Instruct-GGUF/resolve/main/
mistral-7b-instruct.Q4_K_M.gguf
```

⚠️File size: ~4–4.5 GB

---

4. Local System Requirements

Resource	Minimum	Recommended
OS	Linux / macOS / Windows	Ubuntu 22.04
RAM	8 GB	16 GB
CPU	4 cores	8 cores
GPU	Optional	Optional

Resource	Minimum	Recommended
Disk	20 GB	50 GB

---

5. Backend Project Structure (Final)

```
ai-backend/
│
├── models/
│   └── mistral-7b-instruct.Q4_K_M.gguf   ← MODEL FILE
│
├── app/
│   ├── main.py
│   ├── api/
│   │   └── chat.py
│   └── services/
│       ├── llm_service.py
│       ├── retrieval_service.py
│
├── data/
│   └── documents/
│       ├── product.txt
│       ├── faq.txt
│       └── policies.txt
│
├── vector_store/
│   └── faiss_index
│
├── requirements.txt
└── README.md
```

---

6. Installing Dependencies (Local)

Python Version

- Python 3.10 or higher

Install Packages

```
pip install fastapi uvicorn llama-cpp-python faiss-cpu sentence-transformers
numpy
```

`llama-cpp-python` is what loads and runs the GGUF model.

---

7. Loading the GGUF Model in Backend

`llm_service.py`

```
from llama_cpp import Llama

llm = Llama(
    model_path="models/mistral-7b-instruct.Q4_K_M.gguf",
    n_ctx=4096,
    n_threads=8
)

def generate_response(prompt: str) -> str:
    output = llm(
        prompt,
        max_tokens=512,
        temperature=0.3,
        stop=["</s>"]
    )
    return output["choices"][0]["text"].strip()
```

This confirms: - Model file is **inside backend** - Backend fully owns inference

---

8. Making the Bot Learn (RAG – Retrieval Augmented Generation)

8.1 What You Provide

You will provide **plain text documents** describing: - Your product - Features - User flows - FAQs - Policies

Example file: `data/documents/product.txt`

```
The dashboard shows analytics.
Users can export data as CSV using the export button.
```

8.2 Creating Embeddings

Process: 1. Read documents 2. Split into chunks 3. Convert to embeddings 4. Store in FAISS

This is done **once initially** and again whenever documents change.

8.3 Runtime Question Flow

```
User Question
   ↓
FAISS similarity search
   ↓
Relevant document chunks
   ↓
Injected into prompt
   ↓
GGUF model answers
```

This is how the model "learns" safely.

---

9. Prompt Template (Critical)

```
You are a private assistant for this application.
Answer ONLY using the provided context.
If the answer is not found, say you do not know.

Context:
{retrieved_context}

Question:
{user_question}
```

This prevents hallucinations.

---

10. Local API Testing

Start Backend

```
uvicorn app.main:app --reload
```

Test Endpoint

```
POST /chat
{
  "question": "How does export work?"
}
```

Expected: - Correct answer if found in documents - "I don’t know" if not

---

11. Moving From Local to Online Hosting

11.1 Choose a Server

Recommended: - AWS EC2 - DigitalOcean Droplet - Hetzner VPS

Minimum: - 16 GB RAM preferred - Ubuntu 22.04

11.2 Server Setup

```
sudo apt update
sudo apt install python3-pip
```

Upload: - `ai-backend/` - `models/` (GGUF file)

11.3 Install Dependencies on Server

```
pip install -r requirements.txt
```

11.4 Start Backend (Production)

```
uvicorn app.main:app --host 0.0.0.0 --port 8000
```

(Optional) - Use Nginx - Add HTTPS - Use systemd for auto-restart

---

12. Connecting Lovable Website

Frontend Needs Only One Thing

A public HTTPS endpoint:

```
POST https://api.yourdomain.com/chat
```

Flow

```
Lovable UI → Backend API → GGUF Model → Response
```

No model exposure.

---

13. Updating Knowledge (Ongoing Learning)

Whenever content changes: 1. Edit files in `data/documents/` 2. Regenerate FAISS embeddings 3. Restart backend

No retraining required.

---

14. Future Enhancements

- Add GPU for faster inference
- Add memory via vector DB
- Fine-tune via LoRA (advanced)
- Swap GGUF quantization if needed

---

15. Final Confirmation

✔️Model is downloaded by you ✔️Model lives in backend ✔️Works locally ✔️Works online ✔️Learns from documents via RAG ✔️Integrates cleanly with Lovable

This is a **production-grade, correct setup**.

End of Documentation